

# Applications in Multilingual Machine Translation <sup>1</sup>

## Abstract

The CAT2 Machine Translation System, developed in Saarbrücken in 1987, is a natural language application coded entirely in Prolog. Since its initial development, several languages have been implemented on an experimental basis to evaluate the translation methodology, the underlying formalism, the linguistic descriptions, and the effectiveness of the Prolog implementation. Seven years later, it has evolved to the stage where it is now being tested in realistic industrial and academic environments. After briefly describing the current state of the formalism and its implementation, three current applications are described in which CAT2 is being used to perform fully automatic translations: (1) computer system messages between German, English and French, (2) medical diagnoses between Dutch, French and German, and (3) newspaper texts from Spanish to English. An evaluation presents areas still requiring attention. It is maintained that although the CAT2 system is still in a prototype stage, it is achieving a measure of success as a realistic piece of industrial software, at the same time contributing to our understanding of machine translation theory and practice.

**Randall Sharp**  
DGSCA—UNAM  
Apdo Postal 20-059  
04510 Mexico, D.F.  
randy@servidor.unam.mx  
fax: +52-5-622-8540

**Oliver Streiter**  
IAI  
Martin-Luther-Straße 14  
D-66111 Saarbrücken, Germany  
oliver@iai.uni-sb.de  
fax: +49-681-39 74 82

---

<sup>1</sup>In: Proceedings of PAP'95, The Practical Application of Prolog, April 3rd - 6th 1995, Paris.

# Applications in Multilingual Machine Translation

## Summary

This report describes the machine translation system named CAT2, and three of its current on-going applications. The system, developed in Saarbrücken in 1987 as a sideline to the Eurotra Project, is written entirely in Prolog using SICStus Prolog, taking advantage of the *freeze/2* mechanism for delaying the application of linguistic rules pending instantiation.

The linguistic formalism is rule-based, in which the grammar, lexicon and translation rules are maintained separate from the Prolog translation engine, and expressed in a perspicuous user language. The content of the rules is based on features, i.e. attribute-value pairs, thus conforming to standard unification-based grammatical formalisms. In addition, negative, disjunctive and conditional constraints are expressible over features, whose resolution is delayed until sufficient disambiguating information is available.

As in Eurotra, the CAT2 translation methodology is stratification-based; i.e. a syntactico-semantic parse tree is transformed through various levels of representation, ending at the syntactico-semantic tree in the target language. Either a transfer or interlingual structure may form the interface between two languages. We have adopted the transfer methodology, in which the interface structure expresses the predicate-argument relations in the utterance, and thereby resembles in many respects interlingual structures.

The implementation in Prolog is motivated primarily by the advantage of rapid prototyping, since CAT2 is still in a prototype stage and constantly undergoing modifications. In addition, since natural language processing has always been an area of application since the inception of Prolog, a large body of knowledge and experience has developed from which we can draw in our own application.

Many languages have been experimented with using CAT2 since its development. Currently, large grammars and lexicons exist for English, German, Spanish, French and Dutch. Three application areas outlined in the text include (1) the translation of computer system messages from German to English and/or French, (2) the translation of medical diagnoses from Dutch or French to Dutch, French or German, and (3) the translation of newspaper texts from Spanish to English. In the first two cases, the translations are intended to be grammatical and correct; in the latter case, the translation is performed in robust mode, being a rapid word-for-word translation. In all cases, the translation is performed fully automatically, requiring at most human postediting.

An evaluation presents a number of areas in which work is still required. Summarizing, (1) the notion of stratification needs to be reevaluated; (2) the formalism must be extended to handle such phenomena as intersentential references, world knowledge, text processing symbols and punctuation; (3) the usability of the system can be improved by introducing graphic user interfaces and rewriting crucial routines in C for better performance; and (4) the lexical resources need to be expanded considerably.

The CAT2 system is finally being tested in realistic and actual applications. This will result in a validation of the principles — linguistic, translational, and systemic — underlying the system, as well as influence the agenda of improvements to be made.

# Applications in Multilingual Machine Translation

Randall Sharp  
DGSCA—UNAM  
Apdo Postal 20-059  
04510 Mexico, D.F.  
randy@servidor.unam.mx  
fax: +52-5-622-8540

Oliver Streiter  
IAI  
Martin-Luther-Straße 14  
D-66111 Saarbrücken, Germany  
oliver@iai.uni-sb.de  
fax: +49-681-39 74 82

Machine translation has been a favorite application among logic programming specialists working in natural language for quite some time. One of the largest and most ambitious projects, the Eurotra Machine Translation Project, sponsored by the CEC, was implemented primarily in Prolog, in order to take advantage of its rapid prototyping capabilities. Although Eurotra itself did not result in a satisfactory prototype (the project was terminated in 1992), it inspired a methodology which is still being actively pursued today, in the form of the CAT2 machine translation system.

The CAT2 system, whose name derives from the <C,A>,T acronym (“constructors”, “atoms” and “translators”) [Arnold et al. 1986] was based directly on the early Eurotra model, designed for multilingual machine translation. Since its development in 1987 [Sharp 1988] it has been undergoing constant refinement in terms of the formalism, its software implementation, and its linguistic coverage of a variety of languages. It is now reaching a stage where it is being tested — still at a prototype level — in industrial and academic environments, going beyond the experimental toy stage and into true practical applications. This article describes the current efforts being undertaken to validate the CAT2 system — its translation methodology, its formalism, and its implementation — in various real-life applications.

The structure of the report is as follows: the first section describes the CAT2 formalism, i.e. the rule format; the second section briefly describes the Prolog implementation; the third section describes three on-going applications; and the final section reviews the strengths and weaknesses of the CAT2 system in achieving its goal of industrial-strength machine translation.

## 1 The CAT2 Formalism

The CAT2 formalism has been described in a number of articles and papers [Sharp 1988,1991; Sharp and Streiter 1992; Streiter et al. 1994]. Since the formalism is continually evolving, all of these descriptions are in some respects obsolete, although the basic core has remained the same. Here, we look briefly at the formalism as it is today; the final section of the paper discusses areas that will require changing in the future.

### 1.1 Feature Unification

The CAT2 formalism is unification-based, both in the sense of Prolog term unification, and in the sense of a merging of lists of attribute-value pairs. Thus, the following feature bundle:

(1) `{cat=n,agr={per=3}}`

unifies with the following feature bundle:

(2) `{lex=house,string=houses,agr={num=plu}}`

to yield the following:

(3) `{lex=house,string=houses,agr={num=plu,per=3},cat=n}`

but does not unify with the following feature bundle:

(4) `{lex=house,string=houses,cat=v,agr={per=3,num=sing}}`

because of conflicting values for the feature ‘cat’. This type of unification is at the heart of all unification-based grammar formalisms [Johnson 1988; Shieber 1986; Kay 1984]. In addition, negative and disjunctive constraints impose conditions on feature values, which are evaluated “lazily”, i.e. only when sufficient information is present within a feature bundle to unambiguously resolve the constraint. Thus, the following feature bundle:

(5) `{lex=house,string=house,cat=v,agr~={per=3,num=sing}}`

states that the word *house* when used as a verb does *not* carry third person singular agreement. The following disjunctive constraint:

(6) `{lex=house,cat=(n;v)}`

states that the lexeme HOUSE may be either a noun or a verb. The information for a lexeme (indeed for any feature bundle) may be expressed in a hierarchy, such as:

```
(7)  {lex=house}
      >>
      ( {cat=v}
        >>
        ( {string=house}
          >>
          ( {vform=infin}
            ; {vform=fin,tense=pres,agr~={per=3,num=sing}} )
          ; {string=houses,vform=fin,tense=pres,agr={per=3,num=sing}}
          ; {string=housing,vform=presp}
          ; {string=housed}
          >>
          ( {vform=fin,tense=past}
            ; {vform=pastp} ) )
        ; {cat=n,agr={per=3}}
        >>
        ( {string=house,agr={num=sing}}
          ; {string=houses,agr={num=plu}} ) )
```

in which the most general information is expressed at the top, and increasingly specific information is cascaded in alternations under ‘>>’ symbols. Thus, the lexeme HOUSE is either a verb or a noun, and if a verb, then its surface string is either *house*, *houses*, *housing* or *housed*, and if *house*, then either this is the infinitive verb form or a finite form other than third person singular, and if *houses* then it is the finite form for third person singular, and so on.

## 1.2 Rules

Given this feature notation, rules are written to express well-formed lexical and phrasal constituents. All rules in CAT2 describe partial trees. For example, the ubiquitous partial tree:



might be described by the CAT2 rule:<sup>2</sup>

(9) `{cat=s} . [ {cat=np}, {cat=vp} ] .`

in which the symbols ‘.’ and ‘[]’ surround the immediate daughters under the root node, ‘{cat=s}’. A lexical atom has no daughter nodes, and so is described with an empty list under the root:

(10) `{lex=house,string=houses,...} . [] .`

Rules describing a well-formed structure are of two types, identified as **b-rules** and **f-rules**. B-rules define basic structures, as in (9) above. They are used, for example by the parser, in building a parse tree. F-rules essentially validate a structure by enforcing feature unifications over the structure. For example, we might express a rule of subject-verb agreement by writing the following f-rule:

(11) `{ } . [ *, {cat=np}>>{agr=X}, *, {cat=vp}>>{agr=X}, * ] .`

which states that in any structure in which an NP precedes a VP as immediate daughters under an arbitrary root node, their agreement features must unify. The ‘\*’ notation surrounding the NP and VP represent zero or more arbitrary constituents. The ‘>>’ notation, already presented above in (7), is a conditional operator; if the condition on the left side is true, i.e. unifiable, then the consequence on the right side must also be unifiable. If unification of the consequent fails, the rule fails, forcing Prolog backtracking; if the condition fails, then the rule in question is not applicable, and the analysis continues unaltered to the next applicable f-rule.

A given grammar definition consists of a set of b-rules and a set of f-rules. As each b-rule is applied to build some structure, the entire set of f-rules are inspected in order to validate the structure. Thus, we could think of b-rules as “builders” and the f-rules as “filters” (although f-rules may also be used to assign default values to structures by specifying empty consequents). It is an ongoing challenge of applied linguistic engineering to determine the most efficient and effective mixture of builders and filters required to express well-formedness, ranging from thousands of building rules and little or no filtering rules, or a few very general building rules together with a, possibly large, set of filtering rules (c.f. HPSG [Pollard & Sag 1994]).<sup>3</sup>

---

<sup>2</sup>CAT2 rules must include a rule name, used for identifying a rule during debugging. For clarity, rule names are ignored here.

<sup>3</sup>For example, the above f-rule for subject-verb agreement (11) might be better expressed directly in the b-rule in (9), as:

(i) `{cat=s} . [ {cat=np,agr=X}, {cat=vp,agr=X} ] .`

However, if there are many b-rules for S, a single f-rule may better capture the generalization which would otherwise have to be repeated in every such b-rule.

### 1.3 Levels

CAT2, like its Eurotra parent, employs a stratified methodology, in which an analysis undergoes various stages corresponding to levels of representation, as illustrated below:

$$(12) \quad text_s \rightarrow L_1 \rightarrow L_2 \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n \rightarrow text_t$$

A source language text,  $text_s$ , is translated to a target language text,  $text_t$ , by passing through levels  $L_1$  through  $L_n$ , where  $L_1$  is a morphological analysis of  $text_s$  and  $L_n$  is the morphological synthesis generating  $text_t$ ,  $L_2$  and  $L_{n-1}$  are the syntactic levels of the source and target texts, respectively, and between  $L_2$  and  $L_{n-1}$  are zero or more abstract levels, designed to facilitate the transition from source to target structures. In the case of closely related languages, the transformation from  $L_2$  to  $L_{n-1}$  may be direct. In general, however, the syntactic level is not appropriate for translation, so that some interface level(s) is(are) required. In an interlingual approach, a single conceptual level would suffice:

$$(13) \quad text_s \rightarrow L_1 \rightarrow L_2 \rightarrow \dots \rightarrow L_{interlingual} \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n \rightarrow text_t$$

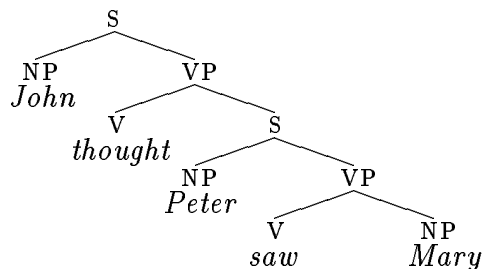
In a transfer-based approach, two intermediate levels, called *interface structures* (IS) in Eurotra, are required:

$$(14) \quad text_s \rightarrow L_1 \rightarrow L_2 \rightarrow \dots \rightarrow L_{IS_s} \rightarrow L_{IS_t} \rightarrow \dots \rightarrow L_{n-1} \rightarrow L_n \rightarrow text_t$$

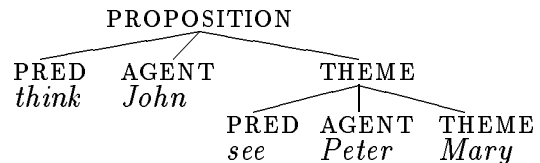
In CAT2, both schemes are possible; the number of levels is arbitrary. For historical and practical reasons, the transfer-based approach has been more highly developed.

The interface structure employed is based on predicate–argument relations within a clause. As a simple example, the sentence “*John thought Peter saw Mary*” has the syntactic structure in (15a) and the interface structure in (15b):

(15) a.

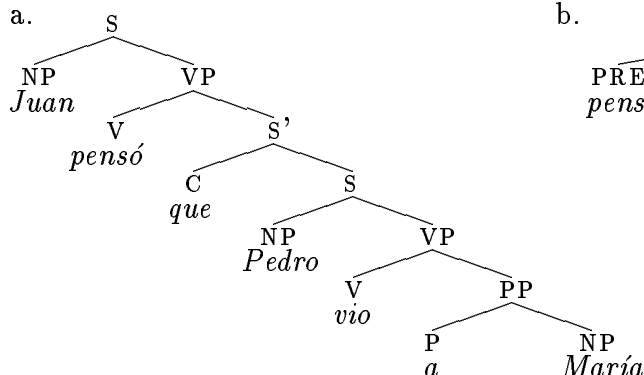


b.

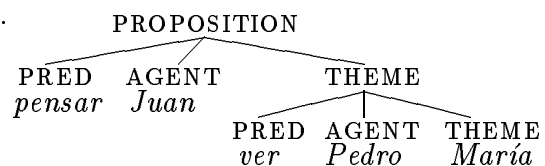


where features on the appropriate nodes of the interface structure carry translationally relevant information as to tense, mood, mode, determination, etc. If translating to Spanish, for example, the interface structure will in fact be isomorphic (16b), while the syntactic structure (16a) would be generated according to the rules of Spanish syntax:

(16) a.



b.



In order to express the relations between levels of representation, CAT2 employs tree-to-tree transformations, consisting formally of **t-rules** for transforming structures and **tf-rules** for transforming (or transferring) features from level to level. A t-rule has the form:

$$(17) \quad \text{LHS} \Leftrightarrow \text{RHS}$$

where LHS and RHS are (partial) tree descriptions at the two adjacent levels. The ' $\Leftrightarrow$ ' symbol expresses the reversibility of t-rules; that is, the LHS is related to the RHS during analysis, in exactly the same way that the RHS is related to the LHS during synthesis. In the event a particular relation is meant to be unidirectional only, a t-rule may also be written as  $\text{LHS} \Rightarrow \text{RHS}$  or  $\text{LHS} \Leftarrow \text{RHS}$ .

An example of a t-rule is the following, which transforms a 3-level phrasal structure into a 2-level structure, and vice versa:

$$(18) \quad \{ \}. [ \text{a}, \{ \}. [ \text{b}, \text{c} ] ] \Leftrightarrow \{ \}. [ \text{b}, \text{c}, \text{a} ] .$$

in which 'a', 'b' and 'c' are markers representing, or binding, subtrees. Each marked subtree is itself transformed by a t-rule, and the resulting transformations are repositioned according to the target side of the t-rule. In this way, a complete source structure is incrementally decomposed into its subcomponents, each of which are transformed by a t-rule, and the resulting transformations are then incrementally composed into a target structure. For example, given the structure in (15a), the left side of the t-rule in (18) will bind the marker 'a' to the subject *John*, 'b' to the verb *thought*, and 'c' to the embedded clause *Peter saw Mary*. A t-rule for each of these substructures must be found, resulting in new bindings which also must be transformed.

The transformation between two languages, the actual transfer stage from  $L_{IS_s}$  to  $L_{IS_t}$ , makes use of a large set of atomic transfer rules, such as:

$$(19) \quad \begin{aligned} \{ \text{lex}=\text{think} \}. [ ] &\Leftrightarrow \{ \text{lex}=\text{pensar} \}. [ ] . \\ \{ \text{lex}=\text{see} \}. [ ] &\Leftrightarrow \{ \text{lex}=\text{ver} \}. [ ] . \\ \{ \text{lex}=\text{house} &\Leftrightarrow \{ \text{lex}=\text{casa} \}. [ ] . \end{aligned}$$

together with a minimum of structural transformation rules. The flexibility of t-rules allows for complex transfer rules, where a single word, e.g. *typewriter*, may have a multiple-word translation, *máquina de escribir*:<sup>4</sup>

$$(20) \quad \{ \text{lex}=\text{typewriter} \}. [ ] \Leftrightarrow \{ \}. [ \{ \text{lex}=\text{maquina} \}, \{ \text{lex}=\text{escribir} \} ] .$$

The preservation of attributes across languages, such as abstractness, animacy, time, and other semantic properties is controlled by tf-rules. For example, the following tf-rule:

$$(21) \quad \{ \} \gg \{ \text{sem}=\text{X} \}. [ ] \Leftrightarrow \{ \} \gg \{ \text{sem}=\text{X} \}. [ ] .$$

ensures that when a lexical subtree is transformed to another lexical subtree, their semantic features must be equal, i.e. unify. Deciding which features to preserve across languages is of course also a matter of ongoing investigation.

Given b-rules, f-rules, t-rules and tf-rules, a complete translation system is possible between any number of languages. Naturally, this brief review ignores many aspects of the formalism (see [Sharp 1994] for a complete description), but is sufficient to indicate the general approach to translation within the formalism.

<sup>4</sup>The nontrivial problem of handling diacritics correctly will not be addressed here.

## 2 Implementation

CAT2 is implemented in SICStus Prolog, taking advantage of the *freeze* construct for blocking the evaluation of goals relating to negative and disjunctive feature unification. It runs on Unix platforms.<sup>5</sup>

The user language for feature bundles illustrated above is compiled into Prolog lists with tail variables. For example, the feature bundle in (1), repeated here as (22a), is compiled into the list in (22b):

- (22) a.        `{cat=n,agr={per=3}}`  
      b.        `[cat=n,agr=[per=3|_|_]`

The unification of two lists is then carried out by merging the lists, using essentially the algorithm presented in [Eisele & Doerre 1986], repeated below:

- (23)    `merge(X,X) :- !.`  
         `merge([A=V1|R1], F2) :-`  
             `delete(A=V2, F2, R2),`  
             `merge(V1, V2),`  
             `merge(R1, R2).`

modified to take into account negative and disjunctive descriptions. Negative constraints, for example, are defined basically by:

- (24)    `merge([A~=V1|R1], F2) :-`  
             `delete(A=V2, F2, R2),`  
             `freeze(V2, negative(V1, V2)),`  
             `merge(R1, R2).`

where `negative/2`, blocked until `V2` is instantiated, ensures that `V1` and `V2` do not unify. Similarly, disjunctive constraints are basically defined by:

- (25)    `merge([A=(B;C)|R1], F2) :-`  
             `delete(A=V2, F2, R2),`  
             `freeze(V2, disjunctive((B;C), V2)),`  
             `merge(R1, R2).`

where `disjunctive/2`, again blocked on `V2`, ensures that `V2` unifies with either `B` or `C`. The actual definitions of `negative` and `disjunctive` are rather complex, since, among other things, they involve inspecting the set of existing blocked goals on `V2`.

CAT2 translation is sentence-based, and is defined by the top-level predicate `translate/4`:

- (26)    `translate(TextS, SL, TextT, TL) :-`  
             `find_translation_path(SL, TL, Path),`  
             `translate(Path, TextS, TextT).`

---

<sup>5</sup>An MSDOS-based version of CAT2 using SICStus is under development.

where `find_translation_path/3` returns a list of level names in `Path` which define the sequence of levels through which the source text `TextS` in the source language `SL` are transformed to the target text `TextT` in the target language `TL`. This list has the form:

(27) [ L1=>L2, L2=>L3, . . . , Ln-1=>Ln ]

The `translate/3` goal then executes the actual translation:

```
(28) translate([],Text,Text).
      translate([L1=>L2|Ls],TextIN,TextOUT):-
          transform(L1,L2,TextIN,TextTMP),
          translate(Ls,TextTMP,TextOUT).
```

where `transform/4` transforms the structure `TextIN` at level `L1` to `TextTMP` at level `L2`. The transformation is either morphological analysis, morphological synthesis, parsing, or a tree-to-tree transformation, depending on the level types of `L1` and `L2`. The `translate/3` predicate is recursively called on the remaining path, until the translation path has been exhausted.

The present implementation of CAT2 has no morphological processor *per se*. That is, no decomposition of an input string into letters and morphemes, nor composition of morphemes into output strings takes place. This entails full-form lexical entries, as illustrated in (7) above. In order to overcome this deficiency, a hook has been inserted into the system to call an external morphological processor, if one exists. In several of our applications (see §3.1 below), we make use of this hook to call the MPRO program [Maas 1994a,b], a Prolog application for analyzing and generating morphological forms for a variety of languages.

The lexicon itself may be stored as internal unit clauses, or, with patch release #9 of SICStus Prolog, as external unit clauses, in separate files. Given the size of our lexicons, this latter approach is now used exclusively. To accommodate the administration of the lexicons, a special database management system has been developed [Sharp & Carl 1994], modelled in part on the SQL standard.

The parser is a pure bottom-up chart-based algorithm, being a modification of the Earley algorithm in which no predictor step is carried out. Each b-rule at the syntactic level is inspected in turn to see if adjacent constituents exist which satisfy the constraints in the body of the b-rule. If so, the b-rule creates a new constituent and the search for applicable b-rules is reinitiated. No start symbol or axiom is defined; the parser returns whatever single constituent has managed to consume all the tokens in the input string, whether it be a sentence, noun phrase, or even a single atomic structure in the case of one-word input. Since no predictions are performed, the parser works best with a minimum of b-rules, typical of approaches utilizing some form of X-bar syntax.

All other processes, including syntactic generation, involve tree-to-tree transformations. The basic strategy is as follows:

1. Find a t-rule whose source side unifies with the current source tree, binding any markers in the t-rule to subtrees.
2. For each marked subtree, perform the transformation from step 1.
3. Construct a provisional target (sub)tree based on the target side of the t-rule, inserting any marked and transformed subtrees in their respective positions as defined in the t-rule.

4. Find a b-rule at the target level which unifies with the provisional target tree.
5. Apply any tf-rules, in order, passing (or transforming) any features from the source tree to the target tree.
6. Apply any f-rules, in order, at the target level to the target tree.

If all the constraints in all four rule types are satisfied, the target tree is complete. If any constraint fails, backtracking returns to the latest choice point, if any; if none, the analysis fails. Even after the last target is reached, the system executes a fail in order to retrieve any additional translations, unless the user has explicitly requested single translations only.

## 2.1 Robustness

In our research, we strive to achieve a high level of linguistic quality in our analyses. However, in real-life texts, especially of the sort described below in §3, it is all too often the case that the analysis will fail, usually during syntactic analysis, but also during transfer to the target language, for example if the translation of a particular word has not (yet) been entered in the transfer dictionary. Rather than failing and simply returning ‘no’, the usual Prolog response to unprovable goals, the CAT2 system enters a robust mode of translation in which the words in the input text are individually translated to the target language, thereby bypassing any syntactic analysis and moving directly to transfer and output. If a given word has no translation, it is passed unaltered to the target text. In this way, CAT2 *always* produces an output. The user may also write special robust translation rules to tailor such translations, improving the quality of the result to such a degree that the “translated” text is hopefully understandable, if not grammatically correct.

In the case of Spanish to/from English, the robust mode often produces astonishingly “good” raw translations, as illustrated below in §3.3, owing primarily to the SVO nature of the two languages. Robust translations involving German or Japanese, however, both considered as SOV languages, produce less intelligible translations. However, it is believed that even these translations may be understandable once the user is accustomed to the behaviour of the system; this needs to be empirically tested. In any event, the goal of the current enterprise is to produce translations fully automatically which require a minimum of postediting.

## 2.2 User Interface

The user interface consists of a command-driven shell. The command language provides a means of loading grammars, inputting source texts and outputting translated texts, and manipulating the objects created during analysis. Objects (i.e. trees) may be transformed, displayed, compared, saved, restored and deleted. A debugging environment is provided to facilitate grammar development.

The grammar definition utilizes the notation illustrated in Section 1 above. In addition, a number of “compiler directives” are provided for defining and delimiting level definitions and rule types, and for including descriptions from files or for dynamically linking a given level definition to another definition.

This linking facility acts like a subroutine facility: two or more level definitions may make use of a common set of rules by dynamically linking to them during analysis, much like a subroutine is called by several procedures. This allows for “grammar sharing”, a concept which not only reduces the overall rule space, but also takes advantage of “universal” properties of languages, or language-family properties, by having these properties defined once and allowing each language, or level definition, to use these descriptions. This also facilitates modifications to the grammars, and in fact benefits from the same advantages of modularization recognized in computer programming theory.

### 3 Applications

The demonstration of CAT2’s adequacy, both as a machine translation methodology and as an implementation in Prolog, is substantiated by the applications to which it is put. In this section, we describe three ongoing projects that utilize CAT2 to perform machine translation. It will be maintained that CAT2 in fact is usable, when taking into account the inherent difficulties of fully automatic machine translation. Nevertheless, recommendations for improvements will be outlined in the final section of the paper.

#### 3.1 Translation of System Messages

Since 1992 the CAT2 system has been tested in the industrial environment of a major software company. This research is being carried out within the CEC-sponsored MLAP project CAT2-EDS, the task of which is to demonstrate the suitability of language processing tools in the face of actual industrial demands<sup>6</sup>. Initially, the domain of the texts to be translated was restricted to system messages which had to be translated from German to English. The inclusion of French as a third language and the complete reversibility of the system has recently been achieved, allowing for six possible translation directions. An extension of the text type to software manuals is currently being investigated.

In order to accommodate the demands of the industrial user, the CAT2 system was extended at its periphery and restyled in its linguistic conception. These modifications and extensions include the following:

1. Having no morphological processor, CAT2 has been modified so as to provide an interface to external morphological processors, in particular the MPRO system [Maas 1994]. English, French and German external morphological components running under SICStus Prolog can now be compiled with the CAT2 system into one Prolog state.
2. Homography reduction is effected within the morphological component, augmenting the efficiency of the context-free parser implemented within CAT2.
3. External lexical databases have been integrated into the system for the storage of huge lexicons. This includes the possibility of compiling terminological and general language lexical databases into a single large working database, applying lexical default rules on

---

<sup>6</sup>Partners in this project are IAI Saarbrücken (coordinator), UMIST Manchester, TALANA Paris and EDS Rüsselsheim.

every lexical item before storing them into the working database. Identical lexical items coming from different terminological databases are stored only once. The application of lexical f-rules guarantees the consistency and completeness of the lexical entries without any effect on the runtime behaviour.

4. With respect to grammars, the monolingual descriptions are composed of a set of well-formedness conditions implemented as f-rules, along with a set of compiler directives for the inclusion of interlingually defined linguistic principles and parameters, available as a library of b- and f-rules. This approach is not only theoretically more satisfying but has proved to be highly efficient for the development and maintenance of the grammars.
5. In search of a suitable IS structure, we elide all functional categories (i.e. articles, prepositions, etc.) and replace them with semantic annotations on the content words affected by the function word. By the same token semantically empty words such as support verbs (e.g. **take a decision**), generic support (*a Japanese **man***) and measure words (*a **piece of information***) are no longer present at IS. Derivationally related words (e.g. *accept, acceptance, acceptable, acceptability*) are represented by the same lexeme (ACCEPT), with each word bearing different semantic descriptions of the temporal, aspectual and modal properties of the item. The target language is then completely free to choose the appropriate functional category, support verb or derivational form in order to express the semantic content expressed in the source language, according to the constraints of the target language.

Examples of successful German to English translations include:

- *Jede Funktion wird im entsprechenden Unterabschnitt ausführlich behandelt.  $\implies$   
Every function is treated in detail in the corresponding subsection.*
- *Anwendungs-Unterschiede zwischen Batch- und Online-Benutzern werden zu gegebener Zeit ausführlicher behandelt.  $\implies$   
Application differences between Batch users and between Online users are treated in greater detail at a given time.*
- *Viele Möglichkeiten sind über die Funktionstasten oder das "Auswahl" -Feld zugänglich.  $\implies$   
Many possibilities are accessible over function keys or over the field "selection".*
- *Die Gesamtanzahl der Anträge für diese Auftragsnummer, die einen "Bezug"-Indikator haben  $\implies$   
The total number of the proposals for this instruction number which have an indicator "cover"*
- *In dieser Phase ist es noch möglich, bereits eingegebene Daten zu ändern.  $\implies$   
In this phase it is still possible to change data which has been input already.*
- *Diese Zusatzmasken werden in Abschnitt "Anwendungsprinzipien" ausführlicher besprochen.  $\implies$   
These additional screens are described in greater detail in the section "application principles".*

### 3.2 Analysis and Translation of Medical Diagnoses

Within the LRE project ANTHEM, sponsored by the CEC, CAT2 is used for the analysis of medical diagnoses expressed in either Dutch or French, with possible output being (a) a translation of the medical diagnosis into either Dutch, French or German, and (b) a semantic representation which can be passed to an expert system for automated coding in ICD<sup>7</sup> (cf. [Ceusters et al. 1994])<sup>8</sup>. Especially the second task makes the ANTHEM API to be developed extremely needed, since standardized data are crucial both for statistical applications and for the indexing of hospital records for storage and retrieval.

Since the medical diagnostic expressions resemble in many respects early child speech (e.g. the nearly complete absence of function words and the weakening of syntactic word-order constraints), the analysis of diagnostic expressions would not be possible without the integration of a semantic model of the domain during (syntactic) parsing. The form of the ANTHEM domain model parallels linguistic structures and can therefore be integrated into the linguistic description. The domain is modeled along three basic syntactico-semantic schemes of composition, each of them represented by a single b-rule: (1) a head-argument structure, (2) a head-modifier structure, and (3) a functional head-complement structure. Every lexical entry is assigned its linguistic properties and an interlingual concept ID, and every concept ID is assigned a semantic ‘type’, as shown here for the Dutch entry *buik* (abdomen):

```
(29)  {string=buik,
       lex='T-D4000',
       type=topo,
       head={cat=n,
            ehead={gen=de,
                  num=sing}}}}
```

The ‘type’ (e.g. *topo*) functions within the domain model as a macro for the concept’s properties with respect to the three basic schemes of composition. The domain model is implemented as a set of lexical f-rules which determine for every ‘type’ the values of the following attributes: (a) **frame**, which describes subcategorization properties, (b) **sem**, which describes semantic properties, (c) **mod**, which describes modifier properties (cf. HPSG [Pollard & Sag 1994]) and (d) **ehead**, which describes the possible head extension of functional categories [Streiter 1994]. After the expansion of the domain model by the f-rules, the same entry has the following form:

---

<sup>7</sup>ICD, the International Classification of Diseases is a system developed by the World Health Organization, and is designed for the classification of morbidity and mortality information for statistical purposes (cf. [WHO 1993]).

<sup>8</sup>The ANTHEM consortium consists of RAMIT Ghent (coordinator), FUNDP Namur, IAI Saarbrücken, CRP-CU Luxembourg, the University of Liège, Datasoft Management nv Oostende and the Military Hospital Brussels.

```
(30)  {string=buik,
       lex='T-D4000',
       type=topo
       head={cat=n,
            ehead={cat=n,
                  gen=de,
                  num=sing
                  sem={location=yes}},
            mod=nil},
       frame={arg1={role=loc,
                   head={ehead={sem={location=yes}}}}}}
```

Paraphrased in natural language, *buik* can enter an argument slot with the selectional restriction `location=yes`, it can modify nothing (`mod=nil`) and it can take one argument which unifies with the description found in `frame={arg1=...}`.

By the joint operation of syntactic and semantic constraints derived from the domain modeling, excellent translations are produced, the quality of which often exceeds that of the source expressions. Here are some examples of Dutch to French and French to Dutch translations:

patella fract li	==>	fracture de la rotule gauche
contusie re enkel	==>	contusion de la cheville droite
spastische bronchitis	==>	bronchite spasmodique
tenniselleboog	==>	epicondylite
tenniselleboog	==>	inflammation dans le coude
lesion menisque interne gauche	==>	letsel aan de interne linker meniscus
luxation recidivante epaule gauche	==>	recidiverende luxatie aan de linker schouder
caries multiples	==>	meervoudige caries
anomalie renale gauche	==>	abnormaliteit aan linker nier

### 3.3 Newspaper Translations

At the Universidad Nacional Autónoma de México (UNAM) in Mexico City, the Academic Computing Center maintains transcriptions of several Mexican newspapers, collected daily from CD-ROMs and available online via the Internet as part of the Gopher service. A current project is to provide at the user's request a translation of any of the available newspaper texts from Spanish into English. This translation is performed offline by CAT2 and sent by email to the user. A future version will perform the translation online, intended to utilize parallel SICStus Prolog (available in 1995) on Sun Sparc 2000s, each with up to 20 coprocessors.

At the time of writing, the Spanish lexicon contains 2818 entries, corresponding to 14,757 word forms; the English lexicon contains 3567 entries, corresponding to 12,620 word forms. (These figures increase daily.)

Because of the general nature of the Spanish texts, ranging from reports on uprisings in the southern state of Chiapas to massacres in the Bosnian conflict to proposed U.S. legislation calling for chemical castration of repeat sexual offenders, the chances of successfully analyzing a given text, whether morphologically, syntactically, semantically and/or translationally, are minimal. The sentences tend to be long (between 20 and 120 words), contain all manner of punctuation, include idiomatic expressions, quotations, foreign words and misspellings, and

generally presuppose a vast store of adult world knowledge. For this reason, the system has been directed to perform all translations in robust mode only, sacrificing quality for speed.

The resulting translations, as expected, require some degree of mental effort to understand, ranging from minimal to outright creative. Nevertheless, they are almost always understandable, which for our application is sufficient. The following texts illustrate the current quality of such robust translations.

For the following typical Spanish text (all diacritics have been removed during a preprocessing stage):

TEHERAN, 1 de marzo (AFP). Dos personas murieron y mas de 20 resultaron heridas en un fuerte sismo registrado el martes de madrugada en varias localidades de la provincia de Fars, en el sur de Iran, anuncio Radio Teheran. El terremoto, de una intensidad de 5.7 grados en la escala de Richter, dano en un 50 por ciento una decena de pueblos entre Firuzabad y Farach-Band, 80 kilometros al sur de Chiraz, capital de la provincia de Fars, segun la radio. El temblor, que duro dos minutos, fue sentido tambien en Chiraz y en otras localidades de la provincia, anadio la radio. Varios temblores secundarios se registraron por la manana, el mas fuerte de ellos alcanzo los 4.8 grados Richter. Al menos una quincena de sismos fueron registrados en los ultimos dias en el sur y sureste de la meseta irani. Un terremoto de 6.6 grados Richter se produjo el pasado miercoles en la provincia de Sistan-Baluchistan (en el sureste) causando seis muertos, varios heridos e importantes danos materiales.

the following is the resulting (unedited) English translation:

TEHERAN, 1 of March (AFP). Two persons died and more of 20 resulted injured in <an/a> strong earthquake registered the Tuesday of morning in various localities of the province of Fars, in the south of Iran, <announced/advertised> Radio Teheran. The earthquake, of <an/a> intensity of 5.7 degrees in the scale of Richter, damage in <an/a> 50 by hundred <an/a> tens of towns between Firuzabad and Farach-Band, 80 kilometers to south of Chiraz, capital of the province of Fars, according to the radio. The tremor, that <lasted/hard> two minutes, <went/was> <sense/felt> also in Chiraz and in other localities of the province, added the radio. Various tremors secondary se registered by the <tomorrow/morning>, the more strong of <they/them> reached the 4.8 degrees Richter. To <less/least> <an/a> fortnight of earthquakes <were/went> registered in the <latter/last> days in the south and southeast of the plateau Iranian. <an/a> earthquake of 6.6 degrees Richter se produced the past Wednesday in the province of Sistan-Baluchistan (in the southeast) causing six dead, various <wounded/injured> and important damages material.

This translation was performed in approximately 23 real-time seconds, running on a Sun Sparc 1000 under Solaris 2.3.

The translation service, now being tested internally and scheduled to go online publicly this year (1995), will be available worldwide via telnet to all users of the UNAM Gopher service.

## 4 Evaluation

The goal of high-quality fully-automatic machine translation has not yet been fully achieved. Nevertheless, the CAT2 project is striving to attain this status. Its formalism stresses simplicity [Sharp & Streiter 1992], and its implementation in Prolog enhances prototype development. It enforces no specific linguistic persuasion; our grammars eclectically combine elements from GB [Chomsky 1986], LFG [Kaplan & Bresnan 1982], and HPSG [Pollard & Sag 1994]. And, as mentioned in the text, all aspects of our approach — the methodology, the formalism, the implementation, and the linguistic descriptions — are undergoing constant change and refinement.

It is perhaps more important to note those areas that still require improvements. We divide this discussion into the four areas mentioned above — the methodology, the formalism, the implementation, and the linguistic descriptions. By systematically attacking each of these areas, we hope to achieve a workable, practical, industrial-quality machine translation system.

### 4.1 Methodology

The current methodology is based primarily on tree-to-tree transformations. It was discovered during the Eurotra period that a proliferation of levels was detrimental to efficient processing. Although of some value conceptually — syntactic relations were kept separate (“autonomous”) from functional relations, which were separate from semantic relations — it not only incurred redundancy in the descriptions, but also led to massive overgeneration, since multiple syntactic structures were generated and then gradually filtered out at subsequent levels. To overcome this situation, we have been able to include syntactic, functional and semantic information within a single lexicon, thereby being available simultaneously during the initial parsing of the utterance.

Nevertheless, we still utilize tree structures at the interface level, with concomitant transformations to and from this level. The interface level provides no new information, it simply rearranges the structure. It is to be investigated if a more direct manner can be found to isolate the translationally relevant information in a source structure, and then generate directly a corresponding target structure on the basis of this information. This would essentially eliminate the concept of multiple levels, each represented as tree structures — probably a desirable consequence.

### 4.2 Formalism

We have stressed the basic simplicity of the formalism, introducing new constructs only when it has been demonstrated that without them the formalism is either incapable or grossly inefficient in its performance. Thus, for example, we have not introduced sets, lists or types into the formalism, as with other experimental approaches to natural language formalisms. This is not to say that we will not at some point introduce these constructs, only that their *necessity* in large-scale systems must first be proven, given the criteria of capability, efficiency and simplicity.

As yet, we operate exclusively in single-sentence mode. Clearly, this is inadequate when dealing with intersentential anaphora. For example, we need to know if the German pronoun

*er* ('he') refers in the target language to a masculine, feminine or neuter entity in order to generate the correct pronoun. At present, the CAT2 formalism is incapable of resolving such intersentential references.

Furthermore, world knowledge is required in order to analyze and disambiguate many texts correctly. The phrase "*pregnant women and children*" is a typical case illustrating the non-associativity of the adjective *pregnant*, as opposed to *various* in "*various women and children*", the knowledge of which can only be attributed to our experience and understanding of the terms *pregnant*, *women*, *children*, etc., and do not rely on linguistic criteria. The collection, representation and utilization of this information will at some point need to be addressed within the present formalism.

Other more mundane areas requiring extensions to the formalism include the treatment of punctuation and word processing controls (e.g. SGML tags). But before these aspects are addressed, the basic formalism must first be validated as to its adequacy and viability for machine translation.

### 4.3 Implementation

The implementation in Prolog has allowed us to develop, extend and change the system with relative ease. This is of course important during prototype development, and will continue to be a priority as long as the system is undergoing fundamental modifications. Once the system — its formalism and methodology — have stabilized, we can consider alternative implementations, e.g. rewriting certain crucial routines in C to achieve better performance.

At the level of the user interface, the current implementation is relatively primitive, based on a text-oriented command language. Extensions involving graphic user interfaces may make the system more user-friendly, again a long-term priority. In addition, the problem of character fonts and alphabets has not yet been resolved, to the dissatisfaction and frustration of virtually all European, Asian and Latin American users. Since our product is languages, this shortcoming affects us most severely.

Another aspect of implementation regards the rule base with which the system performs its translations. As in any large-scale rule-based implementation, the number, nature and complexity of the rules has a bearing on overall performance. Sometimes changing rule orders changes the performance by an order of magnitude, as may changing the order of goals in a Prolog clause definition. A careful application of program profiling may yet uncover sources of inefficient implementation.

### 4.4 Linguistic Description

The rules making up the language descriptions are almost exclusively linguistically based (as opposed to, e.g., statistically based), drawing from the most recent advances in theoretical and computational linguistics. Whether or not machine translation can be effectively achieved on the basis of linguistic knowledge alone has yet to be proved, and in fact is unlikely, given the degree to which extra-linguistic knowledge is called upon in human communication. Nevertheless, we are achieving a degree of success by taking this approach.

The separation of monolingual constraints on well-formedness and an interlingually defined set of principles and parameters which can be included as needed has proved to be most fruitful, as it has allowed for the porting of the grammars described in §3.1 to be used for the application described in §3.2, in which monolingual and interlingual principles are included (or excluded) by the (sub)language at hand. Since the interlingual principles and parameters are mainly syntactic and do not refer to a fixed type of semantic descriptor, the latter can be changed at will in the description of the lexical items (e.g. as selectional restrictions or lexical semantics). Although the three applications presented here use the same syntactic principles (e.g. *head feature principle*, *extended head feature principle*, *subcategorization principle* and *modification principle*) they use quite a different set of semantic descriptors, according to the text type being treated.

Obviously, a large-scale translation application requires large monolingual and bilingual lexicons, and grammars of such stature that they can handle the wide variety of syntactic constructions found in actual texts. This is a long-term process of constant and often tedious refinement. We intentionally restrict text types to strictly informative texts, thereby reducing the range of sublanguages requiring special lexical, grammatical and translational treatment. Nevertheless, successful applications of industrial strength can only be achieved by having huge numbers of accurately coded linguistic resources; we are still at an early stage in this development.

## 5 Conclusion

We have described a machine translation system along with three ongoing major applications. The system is entirely coded in Prolog, and runs on Unix platforms. The applications are medium-scale and growing, both in terms of their rule base and in their coverage. While we are still in a prototype stage of development, we have now left behind the world of toy models and experiments and have advanced to serious applications in industrial environments. We are optimistic about its success.

## References

- Arnold, Doug, Steven Krauwer, Mike Rosner, Louis des Tombe and G.B. Varile (1986) The  $\langle C, A \rangle, T$  Framework in EUROTRA: A Theoretically Committed Notation for MT. Proceedings of COLING'86, Bonn, pp. 297–303.
- Ceusters, Werner, Guy Deville, Emmanuel Herbigniaux, Pierre Mousel, Oliver Streiter and Geert Thiepont (1994) *The ANTHEM Prototype*, IAI Working Paper #31, IAI, Saarbrücken, Germany.
- Chomsky, Noam (1986) **Knowledge of Language**, NY:Praeger.
- Eisele, A. and J. Doerre (1986) A Lexical Functional Grammar System in Prolog. Proceedings of COLING'86, Bonn, pp. 551–553.
- Johnson, Mark (1988) **Attribute–Value Logic and the Theory of Grammar**, CSLI Lecture Notes 16, Stanford:CSLI.

- Kaplan, Ron and Joan Bresnan (1982) *Lexical Functional Grammar: A Formal System for Grammatical Representation*. In: J. Bresnan (ed.), **The Mental Representation of Grammatical Relations**, Cambridge:MIT Press.
- Kay, Martin (1984) *Functional Unification Grammar: A formalism for machine translation*. Proceedings of COLING'84, Stanford, pp. 75–78.
- Maas, Heinz Dieter (1994a) *mpro: Ein System zur Analyse und Synthese deutscher Wörter*. Manuscript, IAI, Saarbrücken.
- (1994b) *Structure of the German morphological dictionary*. Manuscript, IAI, Saarbrücken.
- Pollard, Carl and Ivan Sag (1994) **Head-Driven Phrase Structure Grammar**, Oxford:Oxford Press.
- Sharp, Randall (1994) *CAT2 Reference Manual*, IAI Working Paper #27, IAI, Saarbrücken, Germany.
- (1991) *CAT2: An Experimental Eurotra Alternative*, **Machine Translation 6**, pp. 215–228.
- (1988) *CAT2 – Implementing a Formalism for Multi-Lingual MT*. Proceedings of the 2nd International Conference on Theoretical & Methodological Issues in Machine Translation of Natural Language, Pittsburgh, PA.
- Sharp, Randall and Michael Carl (1994) *A Lexical Database for the CAT2 Machine Translation System*, IAI Working Paper #28, IAI, Saarbrücken, Germany.
- Sharp, Randall and Oliver Streiter (1992) *Simplifying the Complexity of Machine Translation*, **META 37:4**, pp. 681–692.
- Shieber, Stuart (1986) **An Introduction to Unification-Based Approaches to Grammar**, CSLI Lecture Notes 4, Stanford:CSLI.
- Streiter, Oliver (1994) *Komplexe Disjunktion und Erweiterter Kopf: Ein Kontrollmechanismus für die MÜ*, Konvens '94 Tagungsband, 2. Konferenz “Verarbeitung natürlicher Sprache”, Vienna, September 28–30.
- Streiter, Oliver, Randall Sharp, Johann Haller, Catherine Pease and Antje Schmidt-Wigger (1994) *Aspects of a Unification Based Multilingual System for Computer-Aided Translation*, in: Proceedings of the 14th International Avignon Conference–AI'94 (Actes des Quatorzièmes Journées Internationales d'Avignon–IA'94), Volume 3: Natural Language Processing – Le Traitement du Langage Naturel June 1–3, Paris, pp. 111–120.
- WHO (1993) *ICD-10: International Statistical Classification of Diseases and Related Health Problems*, World Health Organization, Geneva.